A Theoretically Grounded Extension of Universal Attacks from the Attacker's Viewpoint

Jordan Patracone, Paul Viallard, Emilie Morvant, Gilles Gasso, Amaury Habrard, Stéphane Canu

A Theoretically Grounded Extension of Universal Attacks from the Attacker's Viewpoint

Jordan Patracone[1,2](✉), Paul Viallard[3], Emilie Morvant[2], Gilles Gasso[4], Amaury Habrard[1,2], and Stéphane Canu[4]

[1] Inria
[2] Université Jean Monnet Saint-Etienne, CNRS, Institut d'Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023, SAINT- ETIENNE, France
[3] Univ Rennes, Inria, CNRS IRISA - UMR 6074, F35000 Rennes, France
[4] Normandie Univ, INSA Rouen UNIROUEN, UNIHAVRE, LITIS, Saint-Etienne-du-Rouvray, France

Abstract. We extend universal attacks by jointly learning a set of perturbations to choose from to maximize the chance of attacking deep neural network models. Specifically, we embrace the attacker's perspective and introduce a theoretical bound quantifying how much the universal perturbations are able to fool a given model on unseen examples. An extension to assert the transferability of universal attacks is also provided. To learn such perturbations, we devise an algorithmic solution with convergence guarantees under Lipschitz continuity assumptions. Moreover, we demonstrate how it can improve the performance of state-of-the-art gradient-based universal perturbation. As evidenced by our experiments, these novel universal perturbations result in more interpretable, diverse, and transferable attacks.

Keywords: Adversarial attacks · Generalization bounds.

1 Introduction

Embedded technologies using artificial Neural Networks (NN) are increasingly present in our daily lives. Their high expressive power has shown great success in various complex tasks [36, 21]. However, since the pioneering work of Szegedy *et al.* [51] that showed the existence of adversarial attacks, some concerns have been raised about the NN's safety and, more particularly, for the safety of the user [24]. The most striking example is that of automated vehicles, where malicious attacks could lead the car to take unwanted action with dramatic consequences [46, 42].

Most of the adversarial attacks are quasi-negligible perturbations that fool the NN prediction. From a fast one-shot method [19] to the first iterative procedures [43, 40, 30, 9, 34], the crafting of adversarial perturbations has lately received a lot of attention from the machine learning community. To this regard, momentum-based methods [16, 54] have shown a promising boost in the transferability of the attacks learned on one NN to other NNs. In addition, various contributions have investigated algorithmic concerns leading to accelerated and

scale-invariant attacks [33] as well as parameter-free attacks [14]. In another line of research, Finlay *et al.* [18] designed attacks exploiting the decision boundary of NNs, while Zhang *et al.* [63] proposed to take into account the structure of images through a principal component analysis. A key particularity of all the above attacks is that they are *specific* (or *example-based*), meaning that they are crafted to attack a *single* example. Therefore, to attack a new example, one needs to learn the associated perturbation once again. Although they are very effective, they have the major drawback of being time-consuming.

On the other end of the spectrum, *universal* (or *example-agnostic*) attacks [62] aim to find an attack that, once learned, can be applied to each new example. Moosavi-Dezfooli *et al.* [39] first showed that there exists a single perturbation, coined universal adversarial perturbation (UAP), which, when added to any new example, is very likely to fool the classifier; a variant exploiting the orientations of the perturbation vectors was proposed [15]. Later, a more efficient method, which relies on a projected gradient descent algorithm was developed [50]. In addition, inspired by the observation that UAP does not attack all classes equally, a class-based universal perturbation was proposed [7]. Although these perturbations are universal, it is hard to interpret why they work on a case-to-case basis. In general, current state-of-the-art universal attacks remain hardly interpretable out-of-the-box and require *a posteriori* tailored studies [61]. These works have suggested that a reasonable assumption is that the perturbations should live in a low-dimensional manifold [22, 52]. Some works proposed solutions to learn such a manifold [27, 63, 4, 23, 56]. Finally, Zhang *et al.* [62] suggested that simple gradient-based UAP methods may lead to better fooling performance.

Contributions. We propose an extension of universal perturbations that bridges the gap between specific and universal perturbations. By combining the best of both worlds, our extension allows us to fool the classifier better than UAP while still being computationally efficient compared to specific attacks. This extension, which we call *generalized universal attacks*, starts from a theoretical observation: we derive a generalization bound on the deviation between the true and the empirical fooling risks of a universal attack. Concretely, we get a bound on how much a learned perturbation is able to fool new examples, confirming that we can use the learned perturbation to attack a model on data coming from the same task. While this bound stands for classical universal perturbation, it can be generalized to a set of universal perturbations. From this theoretical result, given a set of L different *universal* perturbations, we introduce *generalized universal attacks* as follows. The idea is to *specifically* craft an attack for each example by choosing, in an unsupervised manner, a perturbation among a set of L different *universal* perturbations. To do so, we define an optimization problem to jointly learn the L perturbations allowing each example to choose its own perturbation; L can be seen as a tuning parameter controlling the amount of diversity between the perturbations. To solve it, we derive a gradient-based solution with convergence guarantees. We then propose a simple attack procedure. Our experiments confirm the effectiveness of the generalized universal perturbations over previous

gradient-based UAP, in line with the conclusion of Zhang *et al.* [62]. Our results also show that they lead to more interpretable and transferable attacks.

Outline. Section 2 recalls the general framework of adversarial perturbations. In Section 3, after giving a generalization bound for classical universal attacks, we state our main theoretical result, upon which we build our *generalized universal perturbations*. We provide in Section 4 a new method to learn the generalized universal perturbations and an approach for attacking unseen examples based on these learned perturbations. We conduct experiments in Section 5 on benchmark datasets. Note that we defer the proofs to the supplementary material.

2 Preliminaries and related works

We stand in a multiclass setting where $\mathcal{X} \subseteq \mathbb{R}^P$ is a P-dimensional input space and $\mathcal{Y} = \{1, \ldots, c\}$ is the set of $c \in \mathbb{N}_+$ classes. We consider an unknown data distribution D on $\mathcal{X} \times \mathcal{Y}$ that models the task; the associated marginal distribution on $\mathcal{X}$ is $D_{\mathcal{X}}$. We denote D^n, *resp.* $D_{\mathcal{X}}^n$, the distribution of a sample consisting of n data points *i.i.d.* according to D, *resp.* $D_{\mathcal{X}}$.

It is important to note that we are adopting the *attacker's point of view*: as an attacker, we consider that we have at our disposal a *trained model* $f \colon \mathbb{R}^P \to \mathbb{R}^c$ which associates each example $x \in \mathcal{X}$ to its probabilities $f(x) \in \mathbb{R}^c$ to belong to any of the c classes from $\mathcal{Y}$; we denote by $\mathcal{F}$, the set of possible such models. The predicted class of x by f is then defined as $C_f(x) = \operatorname{argmax}_{y \in \mathcal{Y}} f(x)_y$, where $f(x)_y$ represents the y-th output of $f(x)$. We aim to find for each *original example* $x \sim D_{\mathcal{X}}$ a point $a \in \mathcal{X}$ close to x such that $C_f(a) \neq C_f(x)$. Since a is close to x, one might expect f to predict the same class for both examples. Thus, a is called *adversarial example* and said to fool the classifier C_f.

There exists a vast literature for building adversarial examples, measuring their closeness to original examples, and quantifying how much they affect the decision process of f. Here, we embrace the common setting of adversarial example $a = x + \varepsilon$ built by adding an *adversarial perturbation* ε to an original example x. We consider that the two are close if the ℓ_p-norm of the added perturbation is small [31]: $\varepsilon \in \mathcal{B}_p(\delta) = \{e \in \mathbb{R}^P \mid \|e\|_p \leq \delta\}$, for some budget $\delta > 0$.

To measure the discrepancy of the predictions between an original sample and its perturbed version, we consider a loss function $H : \mathbb{R}^c \times \mathcal{Y} \to \mathbb{R}$ taking as inputs $f(a)$ and a class k from $\mathcal{Y}$ (that is either $C_f(x)$ or the original class y of x). In this paper, we use the cross-entropy loss (or its $[0, 1]$-bounded counterpart [17]).

Given the trained model f, and an unlabeled data set $S_{\mathcal{X}} = \{x_i\}_{i=1}^n \sim D_{\mathcal{X}}^n$, the attacker usually crafts adversarial perturbations with two current paradigms: *(i) specific attacks*, where for each $x \in S_{\mathcal{X}}$ we look for a perturbation $\varepsilon(x) \in \mathcal{B}_p(\delta)$, specifically tailored to attack the example x (hereafter, we drop the dependency on x and simply denote ε); *(ii) universal attacks*, where we look for a perturbation ε such that $a = x + \varepsilon$ is an adversarial example for all x from $S_{\mathcal{X}}$. To learn such adversarial perturbations, we assume that we have a labeled sample $S = \{(x_i, y_i)\}_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$ consisting of n data points (where the classes are either the true ones or the ones predicted by the classifier C_f).

We recall now the two most popular specific attacks.

DeepFool [40]. For a given $x \in S_{\mathcal{X}}$, the DeepFool attack is the smallest ℓ_p perturbation managing to fool the classifier C_f. More formally, it solves

$$\text{minimize}_{\varepsilon \in \mathbb{R}^P} \|\varepsilon\|_p \quad \text{s.t.} \quad C_f(x+\varepsilon) \neq C_f(x). \tag{1}$$

PGD [34]. Given S, and a loss function H, the adversarial perturbation for a given $(x, y) \in S$ is defined as the one inside the ℓ_p-ball which maximizes the loss between $x+\varepsilon$ and y. More formally, it solves

$$\text{maximize}_{\varepsilon \in \mathbb{R}^P} \; H(f(x+\varepsilon), y) \quad \text{s.t.} \quad \|\varepsilon\|_p \leq \delta. \tag{2}$$

In practice, the opposite of the objective in Equation (2) is minimized by resorting to a projected gradient method, hence the name of the attack.

On the other side of the spectrum, we recall popular universal attacks.

UAP [39]. The first universal perturbation ε was created iteratively in the following way (until some fooling rate is reached): at each iteration, an example x is selected such that the classifier f is not fooled, *i.e.*, where $C_f(x+\varepsilon) = C_f(x)$, then a perturbation $\Delta\varepsilon$ is crafted with DeepFool by solving Equation (1) to fool the input $x + \varepsilon$; the new perturbation is obtained by updating ε with $\varepsilon \leftarrow \text{Proj}_{\mathcal{B}_p(\delta)}(\varepsilon + \Delta\varepsilon)$, where $\text{Proj}_{\mathcal{B}_p(\delta)}$ is the projection onto $\mathcal{B}_p(\delta)$.

Fast-UAP [15]. This attack is a variant of UAP, where Equation (1) (associated with DeepFool) is replaced from the second iteration by

$$\underset{\Delta\varepsilon \in \mathbb{R}^P}{\text{maximize}} \; \frac{\langle \varepsilon, \Delta\varepsilon \rangle}{\|\varepsilon\|_2 \|\Delta\varepsilon\|_2} \quad \text{s.t.} \quad C_f(x+\varepsilon+\Delta\varepsilon) \neq C_f(x+\varepsilon),$$

where $\langle \cdot, \cdot \rangle$ is the dot product. This new problem aims to find a perturbation $\Delta\varepsilon$ with the closest orientation to the current iterate ε.

UAP-PGD [50]. Given S, the UAP-PGD attack elaborates upon PGD by framing the universal perturbation as the solution of the following problem

$$\underset{\varepsilon \in \mathbb{R}^P}{\text{maximize}} \; \frac{1}{n} \sum_{(x_i, y_i) \in S} H(f(x_i + \varepsilon), y_i) \quad \text{s.t.} \quad \|\varepsilon\|_p \leq \delta. \tag{3}$$

CW-UAP [7]. Recently, UAP-PGD has been extended to class-wise UAP, where a universal perturbation is built for each class. Let $\forall k \in \mathcal{Y}$, $S_k = \{x_i \mid (x_i, k) \in S\}$ be the set of training points of the k-th class, and n_k the size of S_k, then CW-UAP aims at solving

$$\underset{\{\varepsilon_k \in \mathbb{R}^P\}_{k \in \mathcal{Y}}}{\text{maximize}} \; \sum_{k \in \mathcal{Y}} \frac{1}{n_k} \sum_{x_i \in S_k} H(f(x_i + \varepsilon_k), k) \quad \text{s.t.} \quad \forall k \in \mathcal{Y}, \; \|\varepsilon_k\|_p \leq \delta. \tag{4}$$

The solution amounts to learning multiple independent UAP-PGD perturbations, one for each class.

3 Generalization guarantees: we can attack new examples

3.1 From universal perturbation...

In this section, we are interested in giving guarantees on the performance of the learned perturbation in the case of universal perturbations. To do so, we derive below *generalization bounds* (see, *e.g.*, [37]) on the performance of the learned universal perturbation on unseen examples. Formally we are interested in the *quality* of the learned perturbation on the new examples that we measure with the *fooling risk* defined by the following definition.

Definition 1 (Fooling risk for universal perturbation). *Let D be a distribution on $\mathcal{X}\times\mathcal{Y}$, and a labeled set $S\sim D^n$. Given a trained model $f:\mathbb{R}^P\to\mathbb{R}^c$ (learned from a set different from S), and a loss function $H:\mathbb{R}\times\mathcal{Y}\to[0,1]$, the* true fooling risk*, resp. the* empirical fooling risk*, associated to the learned perturbation $\varepsilon\in\mathcal{B}_p(\delta)$ is defined by*

$$R_D^f(\varepsilon)=\mathop{\mathbb{E}}_{(x,y)\sim D} H(f(x+\varepsilon),y), \quad \textit{resp.} \quad R_S^f(\varepsilon)=\frac{1}{n}\sum_{i=1}^{n} H(f(x_i+\varepsilon),y_i).$$

This definition extends the fooling rate, for which H is the 0-1 loss. Since our objective is to learn perturbation that fools the model, our goal is to maximize the risk. To ensure that $R_S^f(\varepsilon)$ is a good estimator of $R_D^f(\varepsilon)$, we prove the following theorem based on the Rademacher complexity [6].

Proposition 1. *For any distribution D on $\mathcal{X}\times\mathcal{Y}$, for any loss function $H:\mathbb{R}^c\times\mathcal{Y}\to[0,1]$, for any model $f:\mathbb{R}^P\to\mathbb{R}^c$, for any budget $\delta>0$, for any ℓ_p-norm with $p\geq 0$ and, for any $\lambda\in(0,1]$, we have*

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall\varepsilon\in\mathcal{B}_p(\delta),\ \left|R_D^f(\varepsilon)-R_S^f(\varepsilon)\right|\leq 2\,\mathfrak{R}_S\left[\mathcal{B}_p(\delta)\right]+3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}}\right)\geq 1-\lambda, \quad (5)$$

where we define the Rademacher complexity [5] of $\mathcal{B}_p(\delta)$ as

$$\mathfrak{R}_S\left[\mathcal{B}_p(\delta)\right]=\mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\sup_{\varepsilon\in\mathcal{B}_p(\delta)}\left[\frac{1}{n}\sum_{i=1}^{n}\sigma_i H(f(x_i+\varepsilon),y_i)\right], \quad (6)$$

where $\boldsymbol{\sigma}=\{\sigma_i\}_{i=1}^n\sim\Sigma^n$ with Σ the Rademacher distribution: $\Sigma(-1)=\frac{1}{2}$, $\Sigma(1)=\frac{1}{2}$.

For a given learned model f, Equation (5) tells that for the perturbation $\varepsilon\in\mathcal{B}_p(\delta)$, the empirical risk $R_S^f(\varepsilon)$ does not deviate too much from the true risk $R_D^f(\varepsilon)$ when the Rademacher complexity $\mathfrak{R}_S[\mathcal{B}_p(\delta)]$ is small. Note that Proposition 1 is valid for any perturbation ε that lives in $\mathcal{B}_p(\delta)$ (whatever the budget δ and the p-norm). As an attacker, we are mostly interested in the lower bound on $R_D^f(\varepsilon)$ that gives an estimate of the "chances" to fool the model. In other words, the more the learned perturbation manages to fool the model f on S, the higher the empirical fooling risk and the higher the chances of fooling the model on

new examples coming from D. The form of the bound of Proposition 1 is quite classical[5], but it has the originality to state a theoretical certification from the attacker's point of view to estimate—given a model f—the true fooling risk $R_D^f(\varepsilon)$ to quantify *how much the learned perturbations are able to fool a given model on unseen examples*. Indeed, in the literature, many recent works aim to understand the generalization abilities in an adversarial setting, but they take the defender's point of view and provide generalization bounds for the so-called *true adversarial risk* that measures *how much the learned model is able to face adversarial attacks on unseen examples*. In other words, while we study the fooling abilities of the perturbations for a given model, those works study the performance (or robustness) of the model when this latter is attacked by adversarial perturbations (without knowing the attack). Among these works, we can mention [58, 26, 3] that are based on an *adversarial* Rademacher complexity of the class $\mathcal{F}$ of the models. Other generalization bounds have been derived for the adversarial risk, such as with VC-dimension [1, 38], covering numbers [41], algorithmic stability [57], perturbation analysis [60], or in PAC-Bayes [53].

3.2 ... to generalized universal perturbations

From the attacker's viewpoint, learning only one perturbation $\varepsilon \in \mathcal{B}_p(\delta)$ may be inefficient: the perturbation may not fool the model for every example in the data set S. More formally, given one perturbation $\varepsilon \in \mathcal{B}_p(\delta)$, the attacker may have difficulties to increase the empirical fooling risk $R_S^f(\varepsilon)$. To increase the chance of fooling the model, we consider $L \in \mathbb{N}_+$ perturbations $\boldsymbol{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_L] \in \mathcal{B}_p(\delta)^L$ and, for each pair $(x_i, y_i) \in S$, we pick one of the L perturbations in $\boldsymbol{\varepsilon}$ which maximizes the loss between $f(x_i+\varepsilon_l)$ and y_i the most. In other words, the perturbations are specific enough for each example while being sufficiently universal to cover all the pairs in S. Hence, considering L perturbations may be better suited to fool all the examples (as shown in Section 5) since it can extend our notion of empirical and true fooling risk as follows.

Definition 2 (Fooling risk for generalized universal perturbations). *Given the assumptions of Definition 1, the* true fooling risk*, resp.* empirical fooling risk*, associated to the learned perturbations* $\boldsymbol{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_L] \in \mathcal{B}_p(\delta)^L$ *is*

$$R_D^f(\boldsymbol{\varepsilon}) = \mathop{\mathbb{E}}_{(x,y)\sim D} \max_{\varepsilon_l \in \boldsymbol{\varepsilon}} H(f(x+\varepsilon_l), y), \quad \textit{resp.} \quad R_S^f(\boldsymbol{\varepsilon}) = \frac{1}{n} \sum_{i=1}^{n} \max_{\varepsilon_l \in \boldsymbol{\varepsilon}} H(f(x_i+\varepsilon_l), y_i).$$

We now extend Proposition 1 to the case of a set of L universal perturbations.

Theorem 1. *Given the assumptions of Proposition 1, for any* $L \in \mathbb{N}_+$*, we have*

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall \boldsymbol{\varepsilon} \in \mathcal{B}_p(\delta),\ \left| R_D^f(\boldsymbol{\varepsilon}) - R_S^f(\boldsymbol{\varepsilon}) \right| \le 2\,\mathfrak{R}_S\left[\mathcal{B}_p(\delta)^L\right] + 3\sqrt{\frac{\ln \frac{4}{\lambda}}{2n}} \right) \ge 1-\lambda, \quad (7)$$

[5] While the form of Proposition 1 is classical, the Rademacher complexity is computed on the set of perturbations $\mathcal{B}_p(\delta)$ rather than on the set of models $\mathcal{F}$.

$$\textit{where} \quad \mathfrak{R}_S\left[\mathcal{B}_p(\delta)^L\right] = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n} \sup_{\boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L} \left[\frac{1}{n}\sum_{i=1}^{n} \sigma_i \max_{\varepsilon_l\in\boldsymbol{\varepsilon}} H(f(x_i+\varepsilon_l), y_i)\right]. \tag{8}$$

Equation (7) is a direct extension of Equation (5): Proposition 1 is a special case of Theorem 1 when $L=1$. Note that this bound is valid for any perturbation $\boldsymbol{\varepsilon}$ that lives in $\mathcal{B}_p(\delta)^L$ with $L\in\mathbb{N}_+$. Importantly, Equation (7) holds for any model f we want to attack that is not necessarily the one we used to learn $\boldsymbol{\varepsilon}$. Hence, the bound holds in the context of adversarial transferability, where attacks are typically learned on simpler surrogate models in view of being used to attack other target models [47]. More formally, given a target model f' whose weights are not available, we aim to select a surrogate model $f\in\mathcal{F}$ in order to craft adversarial perturbations $\boldsymbol{\varepsilon}$ for each example $x\in D_{\mathcal{X}}$ that hopefully also fool the target model f'. In this context, we better consider the learned model $f\in\mathcal{F}$, as shown in the following proposition.

Proposition 2. *Given $\mathcal{F}$ the set of possible models and the assumptions of Theorem 1, then we have*

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\begin{array}{l}\forall f\in\mathcal{F},\, \forall \boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L,\\ \left|R_D^{f'}(\boldsymbol{\varepsilon}) - R_S^{f}(\boldsymbol{\varepsilon})\right| \leq \sup\limits_{\boldsymbol{\varepsilon}'\in\mathcal{B}_p(\delta)^L}\left|R_S^{f'}(\boldsymbol{\varepsilon}') - R_S^{f}(\boldsymbol{\varepsilon})\right| + \sqrt{\dfrac{\ln\frac{2}{\delta}}{2n}}\end{array}\right) \geq 1-\lambda. \tag{9}$$

Equation (9) tells that the empirical risk $R_S^f(\boldsymbol{\varepsilon}')$ of the surrogate model f is representative of the true risk $R_D^{f'}(\boldsymbol{\varepsilon}')$ of a target model f' when $\sup_{\boldsymbol{\varepsilon}'}\left|R_S^{f'}(\boldsymbol{\varepsilon}') - R_S^{f}(\boldsymbol{\varepsilon})\right|$ is small. Intuitively, this term is small if we cannot find a set of perturbations $\boldsymbol{\varepsilon}'$ that differs too much from the perturbations $\boldsymbol{\varepsilon}$ for f'.

4 Optimization and selection of universal perturbations

Motivated by Theorem 1, to learn the $L\in\mathbb{N}_+$ universal adversarial perturbations $\boldsymbol{\varepsilon}=[\varepsilon_1,\ldots,\varepsilon_L]\in\mathcal{B}_p(\delta)^L$, we propose to maximize the empirical fooling risk.

Problem 1. *Let L be the number of universal perturbations to learn. Given $S=\{(x_i,y_i)\}_{i=1}^n$, and the model f, find $\boldsymbol{\varepsilon}=[\varepsilon_1,\ldots,\varepsilon_L]$ solving*

$$\underset{\boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L}{\text{maximize}} \left\{R_S^f(\boldsymbol{\varepsilon}) := \frac{1}{n}\sum_{i=1}^{n} \max_{\varepsilon_l\in\boldsymbol{\varepsilon}} H(f(x_i+\varepsilon_l), y_i)\right\}. \tag{10}$$

When $L=1$ (*i.e.*, for a single perturbation) Equation (10) boils down to Equation (3). In addition, it bears similarities with Equation (4) when L equals the number of classes and each ε_l is independently learned on S_l. Due to the model f, it is worth stressing that Problem 1 is a difficult non-concave maximization problem. Finding its global solution is thus out of reach. To tackle this challenge, we embrace a projected gradient ascent algorithm augmented with an Armijo-like line-search strategy to efficiently find an approximate solution. The corresponding algorithmic procedure is sketched in Algorithm 1 while details are reported in

Algorithm 1 L-UAP

Require: Relaxation parameter $\rho \in]0,1[$
  Initialize $\boldsymbol{\varepsilon}^{(0)} = [\varepsilon_l^{(0)}]_{l=1}^L$
  for $k = 0$ to $K-1$ **do**
    Provide a rough estimate of $\gamma_k > 0$
    Surrogate: $h^{(k)} : \boldsymbol{\varepsilon} \mapsto R_S^f(\boldsymbol{\varepsilon}^{(k)}) + \langle \nabla R_S^f(\boldsymbol{\varepsilon}^{(k)}), \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{(k)} \rangle - (1/2\gamma_k)\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{(k)}\|^2$
    Projected gradient step: $\boldsymbol{\varepsilon}^{(k+1/2)} = \mathrm{Proj}_{\mathcal{B}_{p(\delta)^L}}(\boldsymbol{\varepsilon}^{(k)} + \gamma_k \nabla R_S^f(\boldsymbol{\varepsilon}^{(k)}))$
    $i_k = 0$
    repeat
      $\boldsymbol{\varepsilon}^{(k+1)} = (1-\rho^{i_k})\boldsymbol{\varepsilon}^{(k)} + \rho^{i_k}\boldsymbol{\varepsilon}^{(k+1/2)}$
      $i_k = i_k + 1$
    until $R_S^f(\boldsymbol{\varepsilon}^{(k+1)}) \geq R_S^f(\boldsymbol{\varepsilon}^{(k)}) + \rho^{i_k - 1} h^{(k)}(\boldsymbol{\varepsilon}^{(k+1/2)})$
  end for
  return $\boldsymbol{\varepsilon}^{(K)}$

supplementary material, along with an alternative solution based on a stochastic solver fully exploiting the finite-sum nature of Problem 1. Algorithm 1 comes with convergence guarantees stated below.

Theorem 2 (Convergence [8]). *Let $\{\boldsymbol{\varepsilon}^{(k)}\}_{k\in\mathbb{N}}$ be the sequence of Algorithm 1. Suppose that ∇R_S^f is Lipschitz continuous. Then each limit point of $\{\boldsymbol{\varepsilon}^{(k)}\}_k$ is a stationary point of Problem 1 and $\{R_S^f(\boldsymbol{\varepsilon}^{(k)})\}_k$ converges towards the objective value at the limit point. If R_S^f satisfies the Kurdyka-Łojasiewicz (KŁ) property at any point, then the sequence converges to a stationary point of Problem 1.*

The existence of a Lipschitz constant is crucial to ensure convergence guarantees. Note that studying the Lipschitz continuity of NN and obtaining sharp Lipschitz constant is difficult (*e.g.*, [12, 20]). Although most common loss functions H for classification-based neural networks lack Lipschitz continuous gradients, variants do exist. Hereafter, we resort to one of them, namely the bounded cross-entropy [17] where the probabilities are bounded away from 0 and 1. This loss comes particularly useful as it can be rescaled to yield values in [0, 1] as required by our Theorem 1.

Remark 1. *Many functions met in NNs (e.g., activation functions, loss) are semi-algebraic or tame, and thus, satisfy the KŁ property [2, 59]. Since these concepts are stable under many operations, it is reasonable to assume that many deep NNs f are likely to satisfy the KŁ property and so does R_S^f.*

While little attention is usually devoted to these concerns for crafting adversarial attacks, we show empirically in Section 5 the superiority of the corresponding principled algorithmic solution, even if these assumptions do not always hold.

According to Theorem 1, once $\boldsymbol{\varepsilon}$ has been learned with Algorithm 1 from a sample $S \sim D^n$, we can attack a new example by picking one perturbation among the L perturbations as follows.

Problem 2. *(Attacking unseen example) Given $(x, y) \sim D$, given a NN f, the associated attack is $a = \mathrm{Proj}_{\mathcal{X}}(x + \hat{\varepsilon})$ where $\hat{\varepsilon} = \mathrm{argmax}_{\varepsilon_l \in \boldsymbol{\varepsilon}} H(f(x+\varepsilon_l), y)$. If y is unavailable and assuming a well-performing model f, y is replaced by $C_f(x)$.*

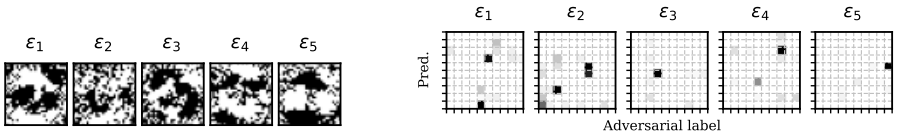


Fig. 1. 5-UAP attacks on MNIST. (Left panel) Learned adversarial perturbations. (Right panel) In the fooling matrix, labels range from 0 to 9 from top to bottom and from left to right.

When f is a NN, in order to evaluate which perturbation from $\boldsymbol{\varepsilon} = [\varepsilon_1, \ldots, \varepsilon_L]$ maximizes the loss function, solving Problem 2 requires performing L independent forward passes through f. Note that since they are independent, they can be performed in parallel to accelerate the computation. We provide below a complexity comparison between specific and (generalized) universal attacks.

Remark 2. *Given a model f that is a neural network whose forward complexity is of $O(d)$ for a single input sample, then the complexity to compute $\nabla H(f(x), y)$ is of order $O(2d)$, since the backward pass is also of order $O(d)$. Then, it follows that* (specific) *for K iterations,* $\text{cost} \sim O(2Kd)$*; and* (generalized-universal) *for L perturbations,* $\text{cost} \sim O(Ld)$*;* (universal) $\text{cost} \sim O(1)$.

Hence, from the standpoint of computational complexity, universal attacks are the most efficient. To a lesser extent, one-shot specific attacks (*i.e.*, $K{=}1$, such as in FGSM [19]) and our proposed generalized universal attack achieve comparable complexity for small L.

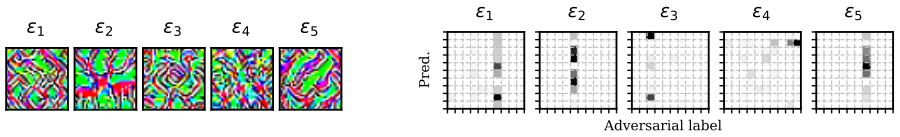
5 Numerical experiments

We now conduct experiments on 3 popular benchmark classification datasets and 2 NN architectures: a differentiable multi-layer perceptron (MNIST [32]) and ResNets (CIFAR-10 [29] and ImageNet). We consider ℓ_∞-attacks with a maximum budget $\delta = 8/255$. For reproducibility purposes, we report implementation details such as pre-processing, data splitting, and model tuning in the supplementary material, as well as results on ℓ_2-attacks.

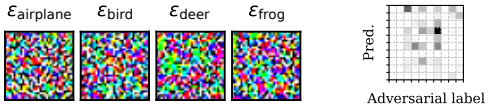
5.1 MNIST Experiments

We begin with the MNIST dataset, useful for interpreting perturbations.

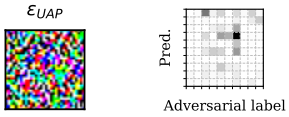
Illustration and role of the universal perturbations. Fig. 1 (left) reports the learned universal perturbations of 5-UAP. Interestingly, they all exhibit strong patterns. In particular, we observe that ε_1 and ε_5 are very similar up to the sign difference. Indeed, since our framework does not handle tuning the sign of the perturbation, two perturbations might be the opposite of each other. It is worth noticing that the universal perturbations learned are consistent throughout multiple splits and random initializations. We report in Fig. 1 (right) the fooling matrices associated to each of the perturbations $\{\varepsilon_l\}_{l=1}^5$. The latter shows the



(a) **5-UAP attacks.**



(b) **CW-UAP attacks.** Each row of the fooling matrix indicates the adversarial label found for each 10 attacks.



(c) **UAP attack.**

Fig. 2. ℓ_∞**-based attacks on CIFAR-10**. In the fooling matrices, labels range {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck} from top to bottom and left to right.

correspondence between the predicted target $C_f(x)$ of some image x (in lines) and the label of the associated adversarial attack (in columns), *i.e.*, $C_f(x+\hat{\varepsilon})$ (see Problem 2). The fooling matrices highlight that each universal perturbation plays a different role. For instance, ε_1 mostly allows to attack images of digits "3" and "9" to be misclassified as "5" and "4", respectively. Instead, ε_3 is mainly used to attack images of "5" into "3". Coincidentally, we can distinguish the tilted digit "3" in ε_3. As opposed to CW-UAP, our UAP attack automatically captures the similarity between multiple digits such as "3" and "9".

Illustration of the behavior of the generalization bound. We report in Fig. 3 an estimation of the lower and upper bounds on $R_D^f(\boldsymbol{\varepsilon})$ from Equation (7). The Rademacher complexity is approximated by resorting to the maximization Algorithm 1 where R_S^f is replaced by the quantity inside the sup of Equation (8). The maximum value of the objective is then averaged over multiple draws of $\boldsymbol{\sigma} \sim \Sigma^n$. As expected, we observe that the empirical fooling risk increases with L. While the Rademacher complexity (and so the generalization gap between $R_D^f(\boldsymbol{\varepsilon})$ and $R_S^f(\boldsymbol{\varepsilon})$) also increases with L, it is interesting to remark that the lower bound on $R_D^f(\boldsymbol{\varepsilon})$ tends to grow, suggesting that considering generalized universal perturbations can increase the chances to fool a model.

Baselines.[6]

We turn to the CIFAR-10 dataset and compare our L-UAP attack with the following universal attacks baselines.

• We compare with UAP-PGD [50] which is closely related to our 1-UAP with a single perturbation, but differs from two aspects. First, the authors have considered a capped loss with parameter β to prevent any single sample from dominat-

[6] Pytorch codes of UAP, Fast-UAP baselines, and our proposed UAP attack will be made publicly available in order to contribute to the *TorchAttacks* repository [28].

ing the objective (hereafter, we use the value $\beta = 9$ that was found to be the best in [50]). Second, the authors resort to the stochastic normalized gradient method ADAM to learn the perturbation. Since their code is not publicly available, we tried to reproduce their version as closely as possible.

• For the sake of consistency, we implemented a Pytorch version of FAST-UAP [15] originally designed for TensorFlow. We use the same hyper-parameters: a desired fooling rate of 80%, a maximum of 10 iterations for DeepFool, and an overshoot of 0.02 to prevent vanishing updates.

• We compare against CW-UAP [7] whose code was granted by the authors.

• We consider standard specific attacks such as FGSM [19] and PGD [34] as well as more advanced techniques, *i.e.*, MI-FGSM [16] and AutoAttack [14], in order to grasp the existing gap of performance between specific and universal attacks. To this effect, we resort to the *TorchAttacks* repository [28].

Illustration & insights about UAP attacks. Similarly to the MNIST experiment, we report in Fig. 2 (a) the learned 5-UAP universal perturbations (left) and their fooling matrices (right). We observe that each UAP universal perturbation plays a different role and illustrates the diversity in perturbations. Indeed, for instance, ε_2 is mostly used to attack images of animals (*bird*, *cat*, *dog*, *frog*, *horse*) so that they become misclassified as *deer*: in ε_2 one can distinguish two deer facing each other. Another example is ε_3 which is mostly used to misclassify images of *airplane* and *ship* as *bird*; in ε_3 one can distinguish a bird. We report in Fig. 2 (b) and (c) the fooling matrices and some universal perturbations of the CW-UAP attack and UAP attack, respectively. Contrary to UAP, we merged all 10 fooling matrices of CW-UAP (one for each class) into a single one since they are all disjointed. Thus, each row of Fig. 2 (b) (left) corresponds to the adversarial label obtained for each of 10 independent class-wise attacks. Unsurprisingly, many of the same couples *(predicted label, adversarial label)* appear in the fooling matrices of UAP, CW-UAP, and UAP. This makes sense since, ultimately, each *(predicted label, adversarial label)* depends on the similarity between image classes and how the classifier proceeds to distinguish between the classes. Despite this resemblance, the key point is that all three methods operate differently. Especially, by its construction L-UAP is able to find an overlapping decomposition of the *(predicted label, adversarial label)* couple. As such, it automatically unveils the similarity between examples belonging to 2 different classes.

Impact of the numbers of training samples. We now take a deeper look at the impact of two parameters on the performance of UAP attacks. More precisely, we study the influence of the number of training samples n and the number of perturbations L (see Problem 1) on the test fooling rate. To this end, we let n and L vary in $\{1\text{K}, 2\text{K}, 3\text{K}, 4\text{K}\}$ and $\{1, 3, 5, 7, 10\}$, respectively. All learned L-UAP attacks are evaluated on a distinct test set. Results, averaged over multiple splits, are reported in Fig. 4. Overall, we observe that increasing L improves the performance, thus confirming that having more perturbations is beneficial to attack f. This observation has to be contrasted with the fact that the amount of data n required to achieve good performance goes in pair with the complexity of Problem 1, hence with L. As such, for $n = 1\text{K}$ or 2K, the

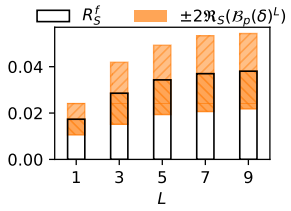


Fig. 3. Generalization bound. The top of the orange bar, *resp.* the bottom, is the approximation of the upper, *resp.* lower, bound of R_D^f.

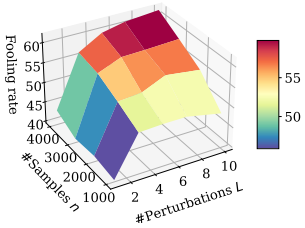


Fig. 4. Impact of parameters. Depending on the number of CIFAR-10 samples, we report the fooling rate of ℓ_∞-based L-UAP attacks.

Table 1. Transferability of ℓ_∞-attacks on a ResNet18 trained on CIFAR-10. Results are split into universal (top), proposed (middle), and specific (bottom) attacks, with bold fonts indicating the highest fooling rate for each target model.

Attack	Source	Transfer			
	ResNet18	ResNet50	MobileNetv2	r-ResNet18	r-ResNet50
UAP-PGD [50]	12.53 ± 0.60	21.51 ± 0.18	39.37 ± 0.19	2.01 ± 0.01	2.54 ± 0.05
FAST-UAP [15]	11.16 ± 1.03	19.65 ± 1.22	36.51 ± 0.30	1.94 ± 0.01	2.33 ± 0.05
CW-UAP [7]	**13.85 ± 0.18**	**21.95 ± 0.28**	**39.62 ± 0.33**	**2.26 ± 0.07**	**2.26 ± 0.06**
1-UAP	36.83 ± 0.93	27.45 ± 0.81	44.11 ± 0.35	2.27 ± 0.02	2.27 ± 0.08
3-UAP	54.03 ± 0.54	28.49 ± 0.56	45.42 ± 0.32	2.55 ± 0.03	2.95 ± 0.08
5-UAP	**55.56 ± 0.57**	**28.87 ± 0.07**	**46.09 ± 0.10**	**2.56 ± 0.05**	**3.10 ± 0.05**
FGSM [19]	53.82 ± 0.00	28.55 ± 0.00	38.10 ± 0.00	**3.02 ± 0.00**	3.14 ± 0.00
MI-FGSM [16]	80.76 ± 0.00	29.95 ± 0.01	35.46 ± 0.01	2.60 ± 0.00	**3.41 ± 0.00**
PGD [34]	**93.61 ± 0.06**	30.47 ± 0.12	**38.17 ± 0.37**	1.94 ± 0.05	2.53 ± 0.08
AutoAttack [14]	93.07 ± 0.00	**31.79 ± 0.16**	38.08 ± 0.27	1.91 ± 0.02	2.41 ± 0.02

performance does not significantly improve (or worse, decrease) with larger L. In what follows, we restrict to a setting made of few samples (*i.e.*, $n = 2$K).

Comparison with baselines. Table 1 reports the performances of ℓ_∞-attacks, in terms of fooling rate. 1-UAP outperforms all universal attacks (UAP-PGD, FAST-UAP, CW-UAP) and, most importantly, it surpasses UAP-PGD, which is closely related. We believe that this is due to our algorithm, which benefits from better optimization guarantees (see supplementary material). In addition, as L grows, we observe an increase in the performance of UAP attacks, thus justifying the advantages of having more degrees of freedom. Interestingly, the UAP attacks manage to improve upon the one-shot specific FGSM attack. However, the performances are still very far behind the more advanced specific attacks. Nonetheless, such differences in performance have to be contrasted with their associated computational complexity (see Remark 2). Overall, L-UAP yields a competitive trade-off between universality and specificity by tuning L.

Transferability of attacks. We now evaluate how the learned attacks on the ResNet18 model manage to fool more complex architectures such as the pre-trained ResNet50 and MobileNetv2 [48] models. We additionally consider two robust models from the *RobustBench* repository [13], namely r-ResNet18 [49] and r-ResNet50 [10], which are trained with some defense mechanisms against ℓ_∞-attacks of budget $\delta = 8/255$. Results are reported in Table 1. Overall, L-UAP systematically yields better transferability than all universal attacks, as shown by the higher fooling rates. Moreover, it manages to outperform specific attacks when the target model architecture is different from the base model on which the attacks have been learned (*i.e.*, Mobilenetv2 vs. ResNet18). Note that this is precisely the setting where most universal attacks also show greater transferability than specific attacks. Surprisingly, although the momentum-based attack MI-FGSM has shown success on ImageNet [16], it does not demonstrate significant transferability on CIFAR10. Interestingly, L-UAP shows competitive results on robust models. It is important to remark that our experimental results are in line with the theory proposed in Proposition 2, which shows that choosing two different architectures may not have an influence on the fooling rate as long as the perturbations similarly fool the two models.

5.2 ImageNet experiments

We tackle a large-scale scenario made of 1K classes. Such a setting is problematic for CW-UAP as computing or storing 1K perturbations exceeds most memory storage spaces: it is not studied here.

Results. We report the results in Table 2. Again, we observe a drastic gap in performance between UAP-PGD and 1-UAP, confirming the superiority of the numerical solution of Algorithm 1 for $L = 1$ over standard UAP-PGD solver [50]. UAP achieves performance of the order of magnitude as specific attacks (*e.g.*, MI-FGSM). It suggests that, for large-scale settings with numerous classes, solely a few universal perturbations are enough to attack most of the images.

6 Conclusion

We have established a theoretical foundation for attackers by deriving a generalization bound that quantifies the effectiveness of universal attacks on new examples and other neural network models. This bound not only applies to classical universal perturbations but also extends to our novel generalized universal perturbations. The latter stands halfway between specific and universal attacks, as evidenced by our numerical experiments. Beyond the gain in performance, generalized universal attacks pull out of existing attacks by capturing meaningful patterns describing the most common flaws to fool the model. We believe that the latter might help to shed some light on how the model operates.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article

Table 2. Performance of ℓ_∞-attacks on a ResNet18 trained on ImageNet. Bold fonts highlight the top fooling rate for each type of attacks.

Attack	ResNet18
UAP-PGD [50]	**27.36 ± 0.00**
FAST-UAP [15]	23.46 ± 0.25

Attack	ResNet18
1-UAP	83.17 ± 2.62
5-UAP	**88.98 ± 1.06**
10-UAP	87.24 ± 1.16

Attack	ResNet18
FGSM [19]	84.53 ± 0.05
MI-FGSM [16]	90.04 ± 0.02
PGD [34]	**94.99 ± 0.06**
AutoAttack [14]	88.23 ± 0.05

Ethic Statement. While focused on DNN attacks, the identified weaknesses could aid in improving their robustness, fostering the development of more reliable DNNs.

References

1. Attias, I., Kontorovich, A., Mansour, Y.: Improved generalization bounds for adversarially robust learning. J. Mach. Learn. Res. **23**(1), 7897–7927 (2022)
2. Attouch, H., Bolte, J., Svaiter, B.F.: Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. Math. Program. **137**(1-2), 91–129 (2011)
3. Awasthi, P., Frank, N., Mohri, M.: Adversarial learning guarantees for linear hypotheses and neural networks. In: ICML (2020)
4. Baluja, S., Fischer, I.: Learning to attack: Adversarial transformation networks. AAAI **32**(1) (2018)
5. Bartlett, P.L., Boucheron, S., Lugosi, G.: Model selection and error estimation. Mach. Learn. **48**(1-3), 85–113 (2002)
6. Bartlett, P.L., Mendelson, S.: Rademacher and gaussian complexities: Risk bounds and structural results. J. Mach. Learn. Res. **3**, 463–482 (2002)
7. Benz, P., Zhang, C., Karjauv, A., Kweon, I.S.: Universal adversarial training with class-wise perturbations. In: IEEE ICME (2021)
8. Bonettini, S., Loris, I., Porta, F., Prato, M., Rebegoldi, S.: On the convergence of a linesearch based proximal-gradient method for nonconvex optimization. Inverse Probl. **33**(5), 055005 (2017)
9. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: IEEE S&P (2017)
10. Chen, T., Liu, S., Chang, S., Cheng, Y., Amini, L., Wang, Z.: Adversarial robustness: From self-supervised pre-training to fine-tuning. In: IEEE/CVF CVPR (2020)
11. Cireşan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. Neural Comput. **22**(12), 3207–3220 (12 2010)
12. Combettes, P.L., Pesquet, J.C.: Lipschitz certificates for layered network structures driven by averaged activation operators. SIAM SIMODS **2**(2), 529–557 (2020)
13. Croce, F., Andriushchenko, M., Sehwag, V., Flammarion, N., Chiang, M., Mittal, P., Hein, M.: Robustbench: a standardized adversarial robustness benchmark. arXiv preprint arXiv:2010.09670 (2020)
14. Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: ICML (2020)

15. Dai, J., Shu, L.: Fast-UAP: An algorithm for expediting universal adversarial perturbation generation using the orientations of perturbation vectors. Neurocomputing **422**, 109–117 (2021)
16. Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., Li, J.: Boosting adversarial attacks with momentum. In: IEEE/CVF CVPR (2018)
17. Dziugaite, G.K., Roy, D.M.: Data-dependent pac-bayes priors via differential privacy. In: NeurIPS (2018)
18. Finlay, C., Pooladian, A.A., Oberman, A.: The logbarrier adversarial attack: making effective use of decision boundary information. In: IEEE/CVF CVPR (2019)
19. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (2015)
20. Gouk, H., Frank, E., Pfahringer, B., Cree, M.J.: Regularisation of neural networks by enforcing lipschitz continuity. Mach. Learn. **110**(2), 393–416 (2021)
21. Grigorescu, S., Trasnea, B., Cocias, T., Macesanu, G.: A survey of deep learning techniques for autonomous driving. J Field Robot **37**(3), 362–386 (2020)
22. Gu, S., Rigazio, L.: Towards deep neural network architectures robust to adversarial examples. In: ICLR, Workshop Track Proceedings (2015)
23. Hayes, J., Danezis, G.: Learning universal adversarial perturbations with generative models. In: IEEE S&P Workshops (2018)
24. Huang, X., Kroening, D., Ruan, W., Sharp, J., Sun, Y., Thamo, E., Wu, M., Yi, X.: A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Comput. Sci. Rev. **37**, 100270 (2020)
25. J. Reddi, S., Sra, S., Poczos, B., Smola, A.J.: Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In: NeurIPS. vol. 29 (2016)
26. Khim, J., Loh, P.L.: Adversarial risk bounds via function transformation. arXiv preprint arXiv:1810.09519 (2018)
27. Khrulkov, V., Oseledets, I.: Art of singular vectors and universal adversarial perturbations. In: IEEE/CVF CVPR (2018)
28. Kim, H.: Torchattacks: A pytorch repository for adversarial attacks. arXiv preprint arXiv:2010.01950 (2020)
29. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. Rep. 0, University of Toronto, Toronto, Ontario (2009)
30. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. In: ICLR Workshop Track Proceedings (2017)
31. Laidlaw, C., Singla, S., Feizi, S.: Perceptual adversarial robustness: Defense against unseen threat models. In: ICLR (2021)
32. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010)
33. Lin, J., Song, C., He, K., Wang, L., Hopcroft, J.E.: Nesterov accelerated gradient and scale invariance for adversarial attacks. In: ICLR (2020)
34. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: ICLR (2018)
35. Mehta, N.A.: Machine learning theory (csc 482a/581a) - lectures 15–18 (2021)
36. Miotto, R., Wang, F., Wang, S., Jiang, X., Dudley, J.T.: Deep learning for healthcare: review, opportunities and challenges. Brief. Bioinform. **19**(6), 1236–1246 (2018)
37. Mohri, M., Rostamizadeh, A., Talwalkar, A.: Foundations of Machine Learning. Adaptive computation and machine learning, MIT Press (2012)
38. Montasser, O., Hanneke, S., Srebro, N.: Vc classes are adversarially robustly learnable, but only improperly. In: COLT (2019)

39. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: IEEE/CVF CVPR (2017)
40. Moosavi-Dezfooli, S., Fawzi, A., Frossard, P.: Deepfool: A simple and accurate method to fool deep neural networks. In: IEEE/CVF CVPR (2016)
41. Mustafa, W., Lei, Y., Kloft, M.: On the generalization analysis of adversarial learning. In: ICML (2022)
42. Nassi, B., Mirsky, Y., Nassi, D., Ben-Netanel, R., Drokin, O., Elovici, Y.: Phantom of the adas: Securing advanced driver-assistance systems from split-second phantom attacks. In: ACM SIGSAC CCS (2020)
43. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: IEEE S&P (2016)
44. Pham, N.H., Nguyen, L.M., Phan, D.T., Tran-Dinh, Q.: Proxsarah: An efficient algorithmic framework for stochastic composite nonconvex optimization. Journal of Machine Learning Research **21**(110), 1–48 (2020)
45. Phan, H.: huyvnphan/pytorch_cifar10 (Jan 2021)
46. Qayyum, A., Usama, M., Qadir, J., Al-Fuqaha, A.: Securing connected amp; autonomous vehicles: Challenges posed by adversarial machine learning and the way forward. IEEE Commun. Surv. **22**(2), 998–1026 (2020)
47. Qin, Z., Fan, Y., Liu, Y., Shen, L., Zhang, Y., Wang, J., Wu, B.: Boosting the transferability of adversarial attacks with reverse adversarial perturbation. In: NeurIPS (2022)
48. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: IEEE/CVF CVPR (2018)
49. Sehwag, V., Mahloujifar, S., Handina, T., Dai, S., Xiang, C., Chiang, M., Mittal, P.: Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In: ICLR (2022)
50. Shafahi, A., Najibi, M., Xu, Z., Dickerson, J., Davis, L.S., Goldstein, T.: Universal adversarial training. AAAI (2020)
51. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: ICLR (2014)
52. Tabacof, P., Valle, E.: Exploring the space of adversarial images. In: IEEE IJCNN (2016)
53. Viallard, P., Vidot, E.G., Habrard, A., Morvant, E.: A pac-bayes analysis of adversarial robustness. NeurIPS (2021)
54. Wang, X., Lin, J., Hu, H., Wang, J., He, K.: Boosting adversarial transferability through enhanced momentum. arXiv preprint arXiv:2103.10609 (2021)
55. Wang, Z., Ji, K., Zhou, Y., Liang, Y., Tarokh, V.: Spiderboost and momentum: Faster stochastic variance reduction algorithms. In: NeurIPS (2019)
56. Xiao, C., Li, B., yan Zhu, J., He, W., Liu, M., Song, D.: Generating adversarial examples with adversarial networks. In: International Joint Conference on Artificial Intelligence (2018)
57. Xing, Y., Song, Q., Cheng, G.: On the algorithmic stability of adversarial training. NeurIPS **34**, 26523–26535 (2021)
58. Yin, D., Kannan, R., Bartlett, P.: Rademacher complexity for adversarially robust generalization. In: ICML (2019)
59. Zeng, J., Lau, T.T.K., Lin, S., Yao, Y.: Global convergence of block coordinate descent in deep learning. In: ICML (2019)
60. Zeng, Y., Shi, Z., Jin, M., Kang, F., Lyu, L., Hsieh, C.J., Jia, R.: Towards robustness certification against universal perturbations. In: ICLR (2023)

61. Zhang, C., Benz, P., Imtiaz, T., Kweon, I.S.: Understanding adversarial examples from the mutual influence of images and perturbations. In: IEEE/CVF CVPR (2020)
62. Zhang, C., Benz, P., Lin, C., Karjauv, A., Wu, J., Kweon, I.S.: A survey on universal adversarial attack. In: IJCAI (2021), Survey Track
63. Zhang, Y., Tian, X., Li, Y., Wang, X., Tao, D.: Principal component adversarial example. IEEE Trans. Image Process **29**, 4804–4815 (2020)

A Algorithmic solutions

In this section, we detail the algorithmic procedures used to learn the generalized universal perturbations.

Notations. For $x \in \mathbb{R}^P$ and $\mathcal{I} \subseteq \{1, \dots, P\}$, $x_{\mathcal{I}}$ stands for the restriction of x to the indices in $\mathcal{I}$.

A.1 Deterministic solver

To maximize R_S^f in Problem 1, we embrace a projected gradient ascent algorithm augmented with an Armijo-like line-search strategy (for ensuring some sufficient increase at each iteration). Its principle is inspired from the minorize-maximization algorithm where, at each step, a lower bound of the empirical fooling risk R_S^f is maximized. Let $\boldsymbol{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_L]$ and some sequence of step-sizes $\{\gamma_k\}_{k \in \mathbb{N}_+}$. Then, at each iteration $k \in \mathbb{N}_+$ the algorithm looks for $\boldsymbol{\varepsilon}^{(k+1/2)} \in \mathcal{B}_p(\delta)^L$ which maximizes the surrogate $h^{(k)}(\boldsymbol{\varepsilon}) = R_S^f(\boldsymbol{\varepsilon}^{(k)}) + \left\langle \nabla R_S^f(\boldsymbol{\varepsilon}^{(k)}), \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{(k)} \right\rangle - (1/2\gamma_k)\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{(k)}\|^2$ with $\langle \cdot, \cdot \rangle$ the Frobenius inner product and $\|\cdot\|$ the induced norm. Such choice is motivated by the fact that, for concave and μ-smooth functions R_S^f, and for all $\gamma_k \leq \frac{1}{\mu}$, we have $R_S^f(\boldsymbol{\varepsilon}) \geq h^{(k)}(\boldsymbol{\varepsilon})$. Henceforth, we have

$$\boldsymbol{\varepsilon}^{(k+1/2)} = \underset{\boldsymbol{\varepsilon} \in \mathcal{B}_p(\delta)^L}{\operatorname{argmax}} h^{(k)}(\boldsymbol{\varepsilon}) = \operatorname{Proj}_{\mathcal{B}_p(\delta)^L}\left(\boldsymbol{\varepsilon}^{(k)} + \gamma_k \nabla R_S^f(\boldsymbol{\varepsilon}^{(k)})\right), \tag{11}$$

which recasts into one projected gradient ascent step. Note that the differentiability of R_S^f depends on the choice H and on the NN f to attack. For instance, for ReLu-based NN, it is likely that $\nabla R_S^f(\boldsymbol{\varepsilon}^{(k)})$ is not well-defined. In that case, and whenever R_S^f is not differentiable, we resort to a sub-gradient instead. We additionally consider a relaxation step of the form $\boldsymbol{\varepsilon}^{(k+1)} = (1 - \alpha_k)\boldsymbol{\varepsilon}^{(k)} + \alpha_k \boldsymbol{\varepsilon}^{(k+1/2)}$, where the relaxation parameter $\alpha_k \in (0, 1]$ is appropriately chosen by an Armijo-like line-search strategy to ensure [8] some sufficient increase in R_S^f. This algorithmic procedure is sketched in Algorithm 1 and referred to as L-UAP. In practice, we suggest initializing the L universal perturbations in a non-informative manner by randomly sampling each $\varepsilon_l^{(0)} \sim [-\delta, \delta]^P$ and additionally projecting onto the ball $\mathcal{B}_p(\delta)$.

About the max operator. Note that R_S^f is first and foremost not differentiable because of the max term of the objective recalled below.

$$\frac{1}{n} \sum_{i=1}^{n} \max_{\varepsilon_l \in \boldsymbol{\varepsilon}} H(f(x_i + \varepsilon_l), y_i). \tag{12}$$

To avoid this concern, one could replace max with a smooth approximation. However, empirically, iterates almost never lie at such singularities. To evidence such finding, we have conducted an additional experiment where the absolute

difference between the two largest values of $\{H(f(x_i+\varepsilon_l), y_i)\}_{\varepsilon_l \in \boldsymbol{\varepsilon}}$ is computed. In particular, we have inspected the smallest absolute difference over the training set, as a function of the iterates $\boldsymbol{\varepsilon}^{(k)}$ of Algorithm 1. We report in Figure 5 one representative run to learn 5-UAP on MNIST (see Section D.1 for details about the experimental setting). At every iteration, the smallest "distance to singularity" is always strictly positive. Hence, the iterates never lie at the discontinuities of the max term.

A.2 Stochastic solver

We propose an additional solver fully exploiting the finite-sum nature of the loss in Problem 1. To this regard, we begin by rewriting it by means of the sample-wise losses r_i, *i.e.*,

$$R_S^f(\varepsilon) = \frac{1}{n}\sum_{i=1}^{n} r_i(\varepsilon), \quad \text{with } r_i(\varepsilon) = \max_{l\in\{1,\dots,L\}} H(f(x_i+\varepsilon_l), y_i).$$

Hereafter, we resort to a stochastic solver based on the well-known variance reduction techniques (see [25, 55, 44]). Since the main computational load comes from the backpropagation through the neural network, we favor the proxSAGA algorithm [25], which does not require an additional loop over multiple epochs. The corresponding algorithmic solution is summarized in Algorithm 2.

Algorithm 2 UAP-ProxSAGA

Initialize $\varepsilon^{(0)} = [\varepsilon_l^{(0)}]_{l=1}^{L}$
Set $g_i = \nabla r_i(\varepsilon^{(0)})$ for every $i \in \{1,\dots,n\}$
Set $\bar{g}^{(0)} = (1/n)\sum_{i=1}^{n} g_i$
for $k = 0$ to $K-1$ **do**
  Instant gradient computation
  Uniformly pick a batch $\mathcal{I}_k \subset \{1,\dots,n\}$ of size b
  $g_{\mathcal{I}_k} = \sum_{i\in\mathcal{I}_k} \nabla r_i(\varepsilon^{(k)})$
  Projected gradient step
  $\alpha^{(k)} = \frac{1}{b}(g_{\mathcal{I}_k} - \tilde{g}_{\mathcal{I}_k}) + \bar{g}^{(k)}$
  $\varepsilon^{(k+1)} = \mathrm{Proj}_{\mathcal{B}_p(\delta)}(\varepsilon^{(k)} + \gamma_k \alpha^{(k)})$
  Updates
  $\bar{g}^{(k+1)} = \frac{1}{n}(g_{\mathcal{I}_k} - \tilde{g}_{\mathcal{I}_k}) + \bar{g}^{(k)}$
  $\tilde{g}_{\mathcal{I}_k} = g_{\mathcal{I}_k}$
end for
return Generalized universal adversarial perturbations $\varepsilon^{(K)}$

Such a solver should become particularly useful in dealing with large datasets by treating one sample at a time. We recall below the convergence guarantees under the assumption of Lipschitz continuity.

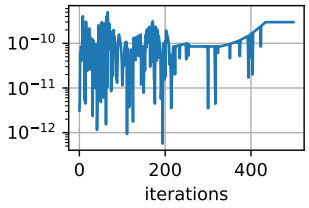


Fig. 5. Distance to singularity.

Theorem 3 ([25]). *Suppose that ∇R_S^f is Lipschitz continuous with Lipschitz constant β. Let $\{\varepsilon^{(k)}\}_{k\in\mathbb{N}}$ be the sequence of Algorithm 2 with fixed step-size $\gamma_k = \gamma \leq 1/(5\beta n)$ and batch-size $b{=}1$. Then, for k uniformly sampled from $\{1,\dots,K\}$, the following holds:*

$$\mathbb{E}\left[\|G_\gamma(\varepsilon^{(k)})\|^2\right] \leq \frac{50\beta n^2}{5n-2}\frac{R_S^f(\varepsilon^\star) - R_S^f(\varepsilon^{(0)})}{K}, \tag{13}$$

where $\varepsilon^\star$ is a maximizer of R_S^f and $G_\gamma\colon \varepsilon \mapsto \gamma^{-1}(\varepsilon - \mathcal{P}_{\mathcal{B}_p(\delta)}(\varepsilon + \gamma\nabla R_S^f(\varepsilon)))$ is the gradient mapping.

Note that Theorem 3 relies on the Lipschitz constant β whose calculation is out of reach. Instead, in practice, we suggest either choosing β large enough or computing a rough estimate at each iteration.

B Proof of Proposition 1 and Theorem 1

The proof of Proposition 1 and Theorem 1 relies on Theorem 3.3 of Mohri *et al.* [37].

Theorem 3.3 of Mohri *et al.* [37]. For any distribution D on $\mathcal{X}\times\mathcal{Y}$, for any set $\mathcal{G}$ of functions $g : \mathcal{X}\times\mathcal{Y} \to [0,1]$, for any $\lambda \in (0,1]$, we have

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall g\in\mathcal{G},\ \mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y) - \frac{1}{n}\sum_{i=1}^n g(x_i,y_i) \right.$$
$$\left. \leq 2\mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\left[\sup_{g'\in\mathcal{G}}\frac{1}{n}\sum_{i=1}^n \sigma_i g'(x_i,y_i)\right] + 3\sqrt{\frac{\ln\frac{2}{\lambda}}{2n}}\right) \geq 1-\lambda. \tag{14}$$

Before proving Proposition 1 and Theorem 1, we recall how we can obtain a two-sided generalization bound from Mohri *et al.* [37]'s Theorem 3.3.

Mohri *et al.* [37]'s Theorem 3.3 brings a one-sided generalization bound, *i.e.*, an upper bound on $\mathbb{E}_{(x,y)\sim D} g(x,y) - \frac{1}{n}\sum_{i=1}^n g(x_i,y_i)$. That being said, Proposition 1 and Theorem 1 provide a two-sided bound, *i.e.*, an upper bound on the

term $\left|\mathbb{E}_{(x,y)\sim D}\, g(x,y) - \frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i)\right|$. One common solution is to use the union bound (*e.g.*, [35]). For the sake of completeness, we state the two-sided bound associated with Theorem 3.3. of Mohri *et al.* [37] in the following lemma.

Lemma 1 (Two-sided generalization bounds). *For any distribution D on $\mathcal{X}\times\mathcal{Y}$, for any set $\mathcal{G}$ of functions $g : \mathcal{X}\times\mathcal{Y}\to[0,1]$, for any $\lambda\in(0,1]$, we have*

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall g\in\mathcal{G},\ \left|\mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y) - \frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i)\right| \right.$$
$$\left. \leq 2\mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\left[\sup_{g'\in\mathcal{G}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i g'(x_i,y_i)\right] + 3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}}\right) \geq 1-\lambda. \quad (15)$$

Proof. We can go through the exact same proof of Mohri *et al.* [37]'s Theorem 3.3 but with $\frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i) - \mathbb{E}_{(x,y)\sim D}\, g(x,y)$ instead of $\mathbb{E}_{(x,y)\sim D}\, g(x,y) - \frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i)$. Hence, we obtain with probability at least $1-\frac{\lambda}{2}$ over $S\sim D^n$

$$\forall g\in\mathcal{G},\quad \frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i) - \mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y)$$
$$\leq 2\mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\left[\sup_{g'\in\mathcal{G}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i g'(x_i,y_i)\right] + 3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}}. \quad (16)$$

Hence, by combining Equations (14) and (16) from a union bound (and with $\lambda/2$ instead of λ), we obtain Equation (15).

We are now ready to prove Proposition 1 and Theorem 1. Note that the proofs are based on Lemma 1.

Proposition 1. *For any distribution D on $\mathcal{X}\times\mathcal{Y}$, for any loss function $H : \mathbb{R}^c\times\mathcal{Y}\to[0,1]$, for any model $f\colon\mathbb{R}^P\to\mathbb{R}^c$, for any budget $\delta>0$, for any ℓ_p-norm with $p\geq 0$ and, for any $\lambda\in(0,1]$, we have*

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall\varepsilon\in\mathcal{B}_p(\delta),\ \left|R_D^f(\varepsilon) - R_S^f(\varepsilon)\right| \leq 2\,\mathfrak{R}_S\left[\mathcal{B}_p(\delta)\right] + 3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}}\right) \geq 1-\lambda, \quad (5)$$

where we define the Rademacher complexity [5] of $\mathcal{B}_p(\delta)$ as

$$\mathfrak{R}_S\left[\mathcal{B}_p(\delta)\right] = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\sup_{\varepsilon\in\mathcal{B}_p(\delta)}\left[\frac{1}{n}\sum_{i=1}^{n}\sigma_i H(f(x_i+\varepsilon),y_i)\right], \quad (6)$$

where $\boldsymbol{\sigma} = \{\sigma_i\}_{i=1}^n\sim\Sigma^n$ with Σ the Rademacher distribution: $\Sigma(-1)=\frac{1}{2}$, $\Sigma(1)=\frac{1}{2}$.

Proof. We will now provide an upper bound of the gap $\left|R_D^f(\varepsilon) - R_S^f(\varepsilon)\right|$. To do so, we define the set of function $\mathcal{G}$ by

$$\mathcal{G} := \left\{ g : (x,y)\mapsto H(f(x+\varepsilon),y) \;\middle|\; \varepsilon\in\mathcal{B}_p(\delta)\right\}.$$

By applying Lemma 1 on the set $\mathcal{G}$, with probability at least $1-\lambda$ over $S \sim D^n$ we have for all $\varepsilon \in \mathcal{B}_p(\delta)$

$$\left|R_D^f(\varepsilon) - R_S^f(\varepsilon)\right| \leq 2 \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n} \sup_{\varepsilon\in\mathcal{B}_p(\delta)} \left[\frac{1}{n}\sum_{i=1}^n \sigma_i H(f(x_i+\varepsilon), y_i)\right] + 3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}},$$

which is the desired result.

We now prove Theorem 1 that has a similar proof than the one of Proposition 1.

Theorem 1. *Given the assumptions of Proposition 1, for any $L\in\mathbb{N}_+$, we have*

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall \boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta),\ \left|R_D^f(\boldsymbol{\varepsilon}) - R_S^f(\boldsymbol{\varepsilon})\right| \leq 2\,\mathfrak{R}_S\left[\mathcal{B}_p(\delta)^L\right] + 3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}}\right) \geq 1-\lambda, \quad (7)$$

$$\textit{where} \quad \mathfrak{R}_S\left[\mathcal{B}_p(\delta)^L\right] = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n} \sup_{\boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L} \left[\frac{1}{n}\sum_{i=1}^n \sigma_i \max_{\varepsilon_l\in\boldsymbol{\varepsilon}} H(f(x_i+\varepsilon_l), y_i)\right]. \quad (8)$$

Proof. We will now provide an upper bound of the gap $\left|R_D^f(\boldsymbol{\varepsilon}) - R_S^f(\boldsymbol{\varepsilon})\right|$. To do so, we define the set of function $\mathcal{G}$ by

$$\mathcal{G} := \left\{g : (x,y) \mapsto \max_{\varepsilon_l\in\boldsymbol{\varepsilon}} H(f(x+\varepsilon_l), y) \;\middle|\; \boldsymbol{\varepsilon} \in \mathcal{B}_p(\delta)^L\right\}.$$

By applying Lemma 1 on the set $\mathcal{G}$, with probability at least $1-\lambda$ over $S \sim D^n$ we have for all $\boldsymbol{\varepsilon} \in \mathcal{B}_p(\delta)^L$

$$\left|R_D^f(\boldsymbol{\varepsilon}) - R_S^f(\boldsymbol{\varepsilon})\right| \leq 2 \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n} \sup_{\boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L} \left[\frac{1}{n}\sum_{i=1}^n \sigma_i \max_{\varepsilon_l\in\boldsymbol{\varepsilon}} H(f(x_i+\varepsilon_l), y_i)\right] + 3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}},$$

which is the desired result.

C Proof of Proposition 2

In order to prove Proposition 2, we first prove the following lemma.

Lemma 2. *For any distribution D on $\mathcal{X}\times\mathcal{Y}$, for any set $\mathcal{G}$ of functions $g : \mathcal{X}\times\mathcal{Y}\to[0,1]$, for any set $\mathcal{G}'$ of functions $g' : \mathcal{X}\times\mathcal{Y}\to[0,1]$, for any $\lambda\in(0,1]$, with probability at least $1-\lambda$ over $S\sim D^n$ we have*

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall S'\in(\mathcal{X}\times\mathcal{Y})^m, g\in\mathcal{G}, g'\in\mathcal{G}', \quad \left|\mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y) - \frac{1}{m}\sum_{i=1}^m g'(x_i',y_i')\right|\right.$$
$$\left. \leq \sup_{g\in\mathcal{G}}\left|\frac{1}{n}\sum_{i=1}^n g(x_i,y_i) - \frac{1}{m}\sum_{i=1}^m g'(x_i',y_i')\right| + \sqrt{\frac{\ln\frac{2}{\delta}}{2n}}\right) \geq 1-\lambda$$

Proof. First of all, we have

$$\sup_{g\in\mathcal{G}}\left(\mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y)\right) = \sup_{g\in\mathcal{G}}\left(\mathop{\mathbb{E}}_{S\sim D^n}\frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i)\right) \leq \mathop{\mathbb{E}}_{S\sim D^n}\sup_{g\in\mathcal{G}}\left(\frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i)\right). \tag{17}$$

Then, from McDiarmid's inequality, we have with probability at least $1-\lambda/2$ over $S\sim D^n$

$$\mathop{\mathbb{E}}_{S\sim D^n}\sup_{g\in\mathcal{G}}\left(\frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i)\right) \leq \sup_{g\in\mathcal{G}}\left(\frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i)\right) + \sqrt{\frac{\ln\frac{2}{\delta}}{2n}}. \tag{18}$$

Hence, by combining Equations (17) and (18), we have

$$\sup_{g\in\mathcal{G}}\left(\mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y)\right) \leq \sup_{g\in\mathcal{G}}\left(\frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i)\right) + \sqrt{\frac{\ln\frac{2}{\delta}}{2n}}. \tag{19}$$

We add $\frac{1}{m}\sum_{i=1}^{m} -g'(x'_i,y'_i)$ to both sides of the inequality to obtain for all $S'\in(\mathcal{X}\times\mathcal{Y})^m$ and $g'\in\mathcal{G}'$

$$\begin{aligned}
&\sup_{g\in\mathcal{G}}\left(\mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y) - \frac{1}{m}\sum_{i=1}^{m} g'(x'_i,y'_i)\right) && (20)\\
&\qquad\leq \sup_{g\in\mathcal{G}}\left(\frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i) - \frac{1}{m}\sum_{i=1}^{m} g'(x'_i,y'_i)\right) + \sqrt{\frac{\ln\frac{2}{\delta}}{2n}}\\
&\qquad\leq \sup_{g\in\mathcal{G}}\left|\frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i) - \frac{1}{m}\sum_{i=1}^{m} g'(x'_i,y'_i)\right| + \sqrt{\frac{\ln\frac{2}{\delta}}{2n}}. && (21)
\end{aligned}$$

We can follow the exact same steps as before, but with $\sup_{g\in\mathcal{G}}(-\mathbb{E}_{(x,y)\sim D} g(x,y))$ and $\frac{1}{m}\sum_{i=1}^{m} g'(x'_i,y'_i)$ instead of $\sup_{g\in\mathcal{G}}(\mathbb{E}_{(x,y)\sim D} g(x,y))$ and $\frac{1}{m}\sum_{i=1}^{m} -g'(x'_i,y'_i)$, to obtain with probability at least $1-\lambda/2$ over $S\sim D^n$

$$\begin{aligned}
&\sup_{g\in\mathcal{G}}\left(\frac{1}{m}\sum_{i=1}^{m} g'(x'_i,y'_i) - \mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y)\right) && (22)\\
&\qquad\leq \sup_{g\in\mathcal{G}}\left(\frac{1}{m}\sum_{i=1}^{m} g'(x'_i,y'_i) - \frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i)\right) + \sqrt{\frac{\ln\frac{2}{\delta}}{2n}}\\
&\qquad\leq \sup_{g\in\mathcal{G}}\left|\frac{1}{m}\sum_{i=1}^{m} g'(x'_i,y'_i) - \frac{1}{n}\sum_{i=1}^{n} g(x_i,y_i)\right| + \sqrt{\frac{\ln\frac{2}{\delta}}{2n}}. && (23)
\end{aligned}$$

Finally, by combining Equations (21) and (23), we have the desired result.

We are now ready to prove Proposition 2.

Proposition 2. *Given $\mathcal{F}$ the set of possible models and the assumptions of Theorem 1, then we have*

$$\underset{S\sim D^n}{\mathbb{P}}\left(\begin{array}{l}\forall f\in\mathcal{F},\ \forall \boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L,\\ \left|R_D^{f'}(\boldsymbol{\varepsilon})-R_S^f(\boldsymbol{\varepsilon})\right|\leq \sup\limits_{\boldsymbol{\varepsilon}'\in\mathcal{B}_p(\delta)^L}\left|R_S^{f'}(\boldsymbol{\varepsilon}')-R_S^f(\boldsymbol{\varepsilon})\right|+\sqrt{\dfrac{\ln\frac{2}{\delta}}{2n}}\end{array}\right)\geq 1-\lambda. \quad (9)$$

Proof. We will now provide an upper bound of the gap $\left|R_D^{f'}(\boldsymbol{\varepsilon})-R_S^f(\boldsymbol{\varepsilon})\right|$. To do so, we apply Lemma 2 with the fixed sets $\mathcal{G}$ and $\mathcal{G}'$ and with $S'=S\sim D^n$. Given $L\in\mathbb{N}_+$, we define the sets of functions $\mathcal{G}$ and $\mathcal{G}'$ by

$$\mathcal{G}:=\left\{g:(x,y)\mapsto\max_{\varepsilon_l\in\boldsymbol{\varepsilon}}H(f'(x+\varepsilon_l),y)\ \middle|\ \boldsymbol{\varepsilon}=[\varepsilon_1,\dots,\varepsilon_L]\in\mathcal{B}_p(\delta)^L\right\},$$

$$\text{and}\quad \mathcal{G}':=\left\{g:(x,y)\mapsto\max_{\varepsilon_l\in\boldsymbol{\varepsilon}}H(f(x+\varepsilon_l),y)\ \middle|\ f\in\mathcal{F},\ \ \boldsymbol{\varepsilon}=[\varepsilon_1,\dots,\varepsilon_L]\in\mathcal{B}_p(\delta)^L\right\}.$$

Hence, by applying Lemma 2, we have with probability at least $1-\lambda$ over $S\sim D^n$

$$\begin{array}{l}\forall f\in\mathcal{F},\boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L,\boldsymbol{\varepsilon}'\in\mathcal{B}_p(\delta)^L,\\ \left|R_D^{f'}(\boldsymbol{\varepsilon}')-R_S^f(\boldsymbol{\varepsilon})\right|\leq\sup\limits_{\boldsymbol{\varepsilon}'\in\mathcal{B}_p(\delta)^L}\left|R_S^{f'}(\boldsymbol{\varepsilon}')-R_S^f(\boldsymbol{\varepsilon})\right|+\sqrt{\dfrac{\ln\frac{2}{\delta}}{2n}}.\end{array}$$

Finally, by setting $\boldsymbol{\varepsilon}'=\boldsymbol{\varepsilon}$, we have the desired result.

D Experimental settings

In this section, we further detail how the numerical experiments were conducted.

D.1 MNIST experiments

Data splitting and pre-processing. The 60K samples of the training set undergo random affine transformations keeping the center invariant. To this effect, we use random rotations between $[11.25,+11.25]$ degrees and a random scaling selected in $[-0.825,+0.825]$. These deformed samples are used to learn f while we randomly pick 500 original un-deformed images from the training dataset to learn the (generalized) universal attacks. The 10K images of the test set are used to evaluate the performance of the attacks. All images are flattened into 784 dimensional rescaled vectors so that the pixel intensity lies within $[0,1]$.

Model to attack. We consider a differentiable model satisfying the KŁ property assumed in Theorem 2 (see Remark 1). To this effect, we resort to the simple multi-layer perceptron from [11], which manages to achieve under 1% test accuracy. It is made of scaled hyperbolic tangent activation functions as well of an input layer, 8 hidden layers and an output linear layer of sizes 784×1000, 1000×1000 and 1000×10, respectively. The network is trained using a stochastic gradient descent with batch size 100 with a learning rate linearly decreasing from 10^{-3} to 10^{-6} over 10^3 epochs.

D.2 CIFAR10 experiments

Data splitting and pre-processing. If not mentioned otherwise, we split CIFAR10 test set into 2K images for learning (generalized) universal perturbations and 8K independent images for evaluating the attacks.

Model to attack. We consider the trained ResNet18 model of Phan [45], which we augmented with an input normalizing layer channel-wise of means $(0.4914, 0.4822, 0.4465)$ and standard deviations $(0.2471, 0.2435, 0.2616)$.

D.3 ImageNet experiments

Data splitting and pre-processing. We resort to the popular ILSVRC2012 validation subset of the ImageNet dataset. The 50k images are randomly split into two halves. The first half is used to learn the universal perturbations while the second half is regarded as test set to evaluate the attacks. All images are resized into 256×256 followed by a cropping of size 224×224 around the center and a rescaling of the pixels intensity into $[0, 1]$. Results are averaged over 5 splits.

Model to attack. We analyze a pretrained ResNet18 model from the Torchvision library augmented with a normalizing layer of mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$ achieving a test accuracy of 69.76%.

L-UAP solver. Contrary to the previous experiments, we consider the Prox-SAGA solver of Algorithm 2 in order to learn the L-UAP perturbations. The step-size and batch-size are set to $\gamma = 0.05$ and $b = 1$, respectively.

E Additional results

In the next sections, we provide complementary results on both MNIST and CIFAR10 datasets.

E.1 MNIST experiments

We analyze the training behavior of L-UAP attacks with $L \in \{1, 3, 5, 10\}$ universal perturbations learned with the Algorithm 1. The experiment is repeated over 5 independent seeds and the averaged training loss is reported in Figure 6. Independently of L, it shows the well-behaved increasing behavior of the loss along the number of epochs. In addition, it supports the fact that having more universal perturbations does permit the achievement of higher dissimilarity, hence higher loss values. This is seconded by the mean test fooling rate reported for each of the L-UAP attacks since we observe an increased fooling rate as L grows. On a side note, on this simple dataset, it is difficult to fool the studied network f, hence justifying the small fooling rates depicted in Figure 6.

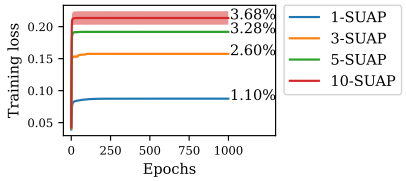


Fig. 6. Training behavior of ℓ_∞-based L-UAP attacks on MNIST. The averaged training loss is reported for 1, 3, 5 and 10 universal perturbations along with the associated test fooling rate.

E.2 CIFAR10 experiments

Comparison between ℓ_∞ and ℓ_2 attacks. We report in Table 3 the performance comparison of both ℓ_∞ and ℓ_2-attacks with a maximum allowable budget of $\delta = 8/255$ and $\delta = 0.5$, respectively.

Comparison of UAP solvers for $L = 1$ perturbation. We compared our proposed deterministic solver from Algorithm 1 with the stochastic solvers advocated in [50]), namely SGD and Adam with a batch size of 128. All three solvers are initialized identically and are run until convergence is reached. Learning rates are cross-validated in a fine logarithmic grid between $10^{-1.5}$ and 10^3. To eliminate any potential influence of our different loss function on the observed performance improvement, we thoroughly examined all three loss functions H. More precisely, we have considered the cross-entropy (*ce*), the bounded cross entropy (*bce*) [17] and the capped cross entropy (*max-ce*) introduced by the authors of the UAP-PGD attack [50]). The test fooling rate, averaged over multiple data splits, are reported in Figure 7. Interestingly, the choice of loss does not significantly impact the results. In addition, both SGD and Adam yield similar performance, while our deterministic solver substantially improves the quality of the learned attack.

Table 3. Performance of attacks on a ResNet18 trained on CIFAR-10. Bold fonts highlight the best fooling rate in universal (top), sproposed (middle) and specific (bottom) attacks.

Attack	ℓ_∞-fooling rate (%)	ℓ_2-fooling rate (%)
UAP-PGD [50]	12.53 (± 0.60)	2.67 (± 0.21)
FAST-UAP [15]	11.16 (± 1.03)	2.53 (± 0.19)
CW-UAP [7]	**13.85 (± 0.18)**	**2.77 (± 0.09)**
1-UAP	36.83 (± 0.93)	3.43 (± 0.26)
3-UAP	54.03 (± 0.54)	4.93 (± 0.50)
5-UAP	**55.56 (± 0.57)**	**7.09 (± 1.22)**
FGSM [19]	53.82 (± 0.00)	N/A
MI-FGSM [16]	80.76 (± 0.00)	N/A
PGD [34]	**93.61 (± 0.06)**	89.23 (± 0.02)
AutoAttack [14]	93.07 (± 0.00)	**92.41 (± 0.01)**

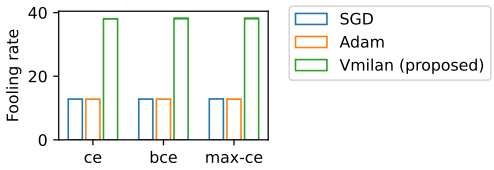


Fig. 7. Comparison of solvers for $L = 1$. The averaged test fooling rate is reported when learning a single perturbation with multiple solvers and different type of losses H. We have considered the cross entropy (ce), its bounded variant (bce) and its clamped version (max-bce).